Cari Benesh

Michelle Ruse

Data Structures

5/2/2018

Final Project: File Read/Sort

My final project submission is a program that reads a data file into an ArrayList object and then sorts the data by a specified parameter. In this case, the file contained vendor sales data for all the stores in a chain. Sorting by the gross profit column of the report generates store rankings for each vendor. Other filters can be added to return results for all vendors by store, or for all stores by a specified vendor, as needed.

Three excel files were included with the project: salesData.csv for all vendors and all stores for the RankData class, storeData.csv with storeIDs and District Names for the StoreData class, and vendorData.csv with vendor IDs and names to be used with the VendorData class. The RankData class holds store numbers, vendorIDs, and the gross profit amounts. StoreData and VendorData are included to be used as filters: each class has a get() method that allows you pull an ID out to match against the store or vendor ID in the RankData class so you can limit the return results.

Each of the three Data classes is used to create a corresponding ArrayList as the appropriate files are read: storeList, vendorList, and rankList. RankList is the only list that is sorted, but with some additional coding the vendors could be sorted by vendorType and stores could be sorted by districtName. I used the Java library’s Collections.sort() method and Comparative method to sort the rankList in descending order of gross profit. I also attempted, in the SortedList class, to create a quickSort() method to use in place of the prebuilt sort methods, but I still have work to do to implement it.

Time was one of my bigger blockers for this project. I did not keep time logs and found it fairly challenging to find enough time to get started while balancing the regular homework assignments. Initially I spent about two-three hours a week researching and trying to settle on how to organize the different structures. Over the past two weeks I spent closer to twelve hours a week hashing out problems and researching solutions. The bulk of the progress definitely took place after I was able to meet briefly with the instructor face to face and describe the problem. I did end up making changes to the structure after we spoke, but that is the point that the structure finally clicked in my head.

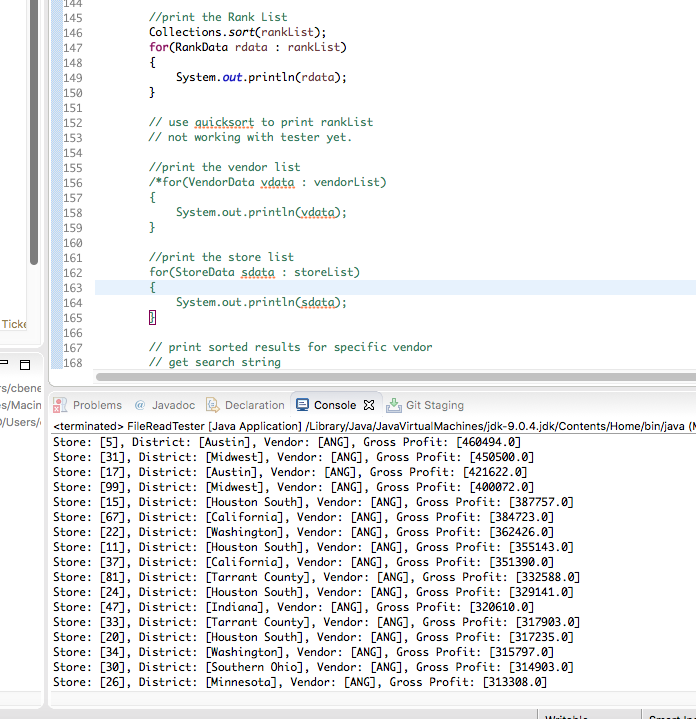
The scope of my project remained similar to my initial proposal, but the way I imagined putting it together was fuzzy at the outset. I imagined using a Rank class as a Link object to connect the Store and Vendor classes and putting it all in a Linked List that I could sort. It was only after getting the file reading to work that I realized I had no reason to artificially split the store and vendor information up, since it is all contained in a single line. At that point I realized that if I could develop a GUI for the project, the storeList and vendorList I created could be used to populate a drop down list that would allow the user to make selections rather than reading a file and trying to copy values into a console. I did not have enough time left to work on the GUI for this project but I do plan to keep working on it for myself.

Developing my own sort method and the GUI are the two big pieces of this project I would like to get to. I would also like this program to write to a file. These are two goals I feel confident I can achieve with more time. For now, the interface is very basic. At this point the FileReadTester.java file does all of the work and does not require user input. I have commented out several print methods (one for the vendorList, one for the storeList, and one filtering the rankList by a vendorID). Each of those can be run individually. The program is currently set to run the full rankList, sorted by gross profit. I do get an error message about an input string, but have not been able to find the issue. It does not affect the returned data.

The principles of good programming are as follows:

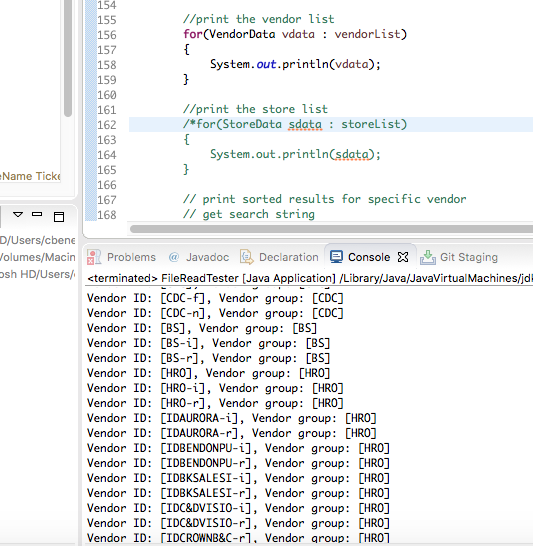
1. Modularity: design with portability in mind. Use classes so code can be reused, and encapsulate methods/data to maintain integrity.
2. Efficienty: know the cost in time and memory for the data structures you use, as well as the project requirements, so you can make the best choice for that project.
3. Robustness: build in error handling. Always assume the user will make a mistake.
4. Usability: a program should be designed with a specific purpose in mind and achieve that purpose. Functionality that doesn’t add to (or actually detracts) from the main purpose should be dropped. Keep the program and interface user-friendly.
5. Should be readable: write self-documenting code by using explanatory variable names, following standard spacing/naming conventions, and write your comments as if you’ve never seen the code before.
6. Elegant: keep the code concise, readable, and efficient.

USER MANUAL:

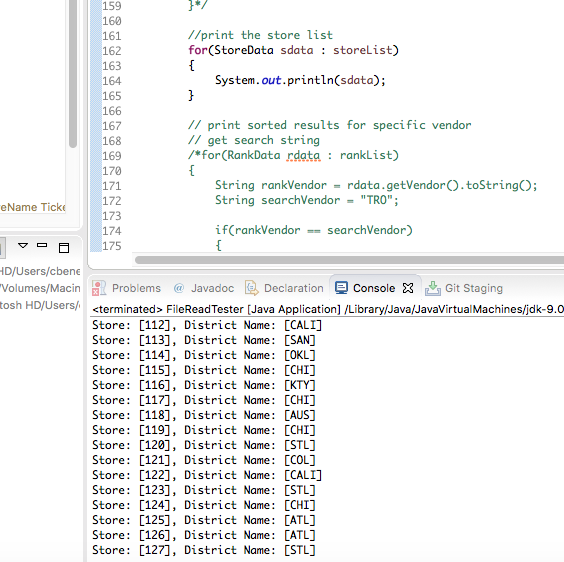
ScreenShot #1:

To print the full sorted rankList, activate lines 146-150 and comment out the rest.

Screen Shot #2



To print the vendorList, activate lines 156 – 159 and comment out the other print sections.

ScreenShot #3

To print the storeList, activate lines 162-165 and comment out the remaining print sections.